Interactive Exploration of Medical Images on the Grid

Cécile Germain-Renaud[1] Romain Texier[2] Angel Osorio[3]

[1]LAL CNRS-Université Paris-Sud [2]PCRI -Université Paris-Sud [3]LIMSI CNRS-Université Paris-Sud

1. Introduction

Many clinical medical imaging applications combine the need for high-end computing and data storage requirements and the need for human supervision. For instance, classical problems such as segmentation or registration do require high-end parallel computers, and do not have an all-hands fully reliable automatic solution; this difficulty can be alleviated in admitting some degree of user interaction. The health care challenge is to transfer experimental research first to clinical practice, then to routine clinical practice. Combining the medical user expertise and the resource of the grid in compute and data intensive tasks is a promising way to tackle this challenge. More generally, the massive use of computing power today is interactive desktop computing. Can the grid metaphor apply not only to factories (large virtual organizations), but to light bulbs (the individual interactive user), when the power of the grid is required?

Grid-enabling interactive applications raises many issues. Two of them are addressed in this paper: first, the cost of grid protocols; second, resource sharing between interactive users and heavyweight computations. We exemplify our discussion with the PTM3D interactive application developed at LIMSI for visualizing, navigating through, and analyzing complex medical image data sets [1]. In this paper, we show that PTM3D can take advantage of the Grid computing power for costly reconstruction algorithms. These computations involve multiple components that can benefit from fine grain scheduling, and require user steering.

Section 2 sketches some issues related to the distribution of interactive exploration of medical images. Section 3 deals with the dynamic parallelization of the most time-consuming algorithm of PTM3D. Section 4 shows how the user-level scheduler defined in the previous section can be deployed in a Globus framework. Section 5 explores the resource sharing issue.

2. Objectives

2.1. The visualization pipeline

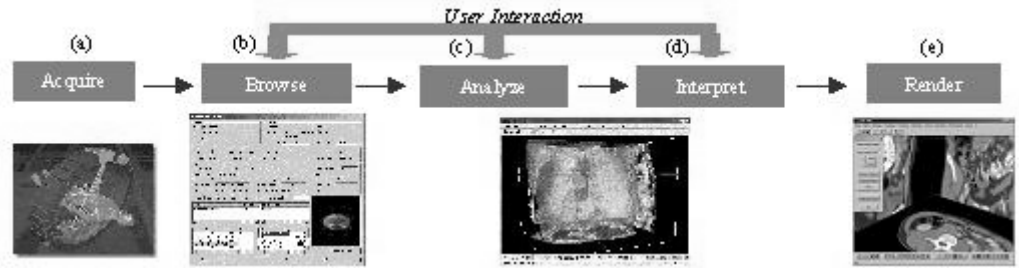

Figure 1: The visualization pipeline

The steps of a visualization-based interaction are represented fig. 1. The four last steps can be executed remotely, and human interaction is present at all steps. Data comes from Scanner, a MRI instrument or a PET-scan, and data and metadata are further encoded in DICOM format. Browsing is the graphical navigation through the metadata. Analysis translates the physical data to data relevant to the user request. This task can be simple, such as the planar projections used when navigating a radiological image, or very time-consuming, e.g. in automatic segmentation, multi-modal fusion or registration. Interpretation generates a visualizable representation of the data. In volume reconstruction, this is the most costly step. For many cases of clinical practice, a high-end PC provides the necessary graphical performance for the rendering/rasterization step.

2.2. Scenarios

The easiest scenario for Grid use is Local Data Remote Computation (LDRC). Data are localized near the user, on his own PC or a local network. All preprocessing can thus be local, and only the most compute- intensive parts, analysis or interpretation, are candidate to remote execution. Even in this relatively simple case, grid computing can be the optimal choice. A more complex scenario is Remote Data Remote Computation (RDRC). The DICOM data are stored remotely in a central or distributed database and may as well be replicated (Globus [5]) or cached (DPSS/IBP [6]). Steps (b) (c) and (d) in fig. 1 are then either local or remote, while the user is supposed to interact through a local view (step (e)). Our current work focuses on LDRC. A general discussion of the issues related to RDRC, along with other issues related with grid medical imaging, can be found in the report of the first IMAGE meeting [7].

To be concrete, we exemplify LDRC scenario on PTM3D. PTM3D features a sophisticated volume reconstruction and measurement tool based on semi-automatic segmentation. Volume measurement targets *diagnostic or therapy*

planning. For large organs (e.g. lung), the sequential time is not in the interactive range. Volume reconstruction also targets *augmented reality*: supplementing real data from some viewpoint, source, date or modality with real data from another such item. Fig. 2 shows augmented reality applied to percutaneous nephrolithotomy. For this intervention, the patient must be ventrally positioned; to figure out the kidney shape and position, which might differ broadly from the scanner ones (dorsal position), the anatomic marks (iliac bones, ribs, rachis) in the scanner and on the live patient are matched, through a video projection of the PTM3D output.

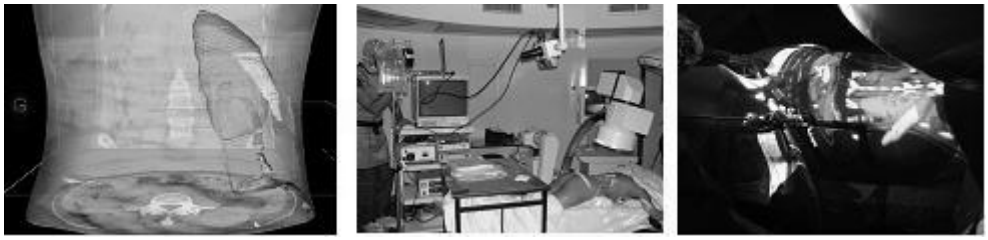

Figure 2: Augmented reality in surgical planning for percutaneous nephrolitotomy

Surgical planning for all these interventions is a complex process requiring days. The volume reconstruction part is thus quite modest, but in the order of half an hour for sequential execution. Interactive (less than 2 minutes) response time for will allow for more accurate measurements, and has thus the potential to improve health care. Many other compute-intensive applications in this area are amenable to efficient parallelization [2, 3, 4].
The PTM3D 3D-segmentation and volume reconstruction algorithms exemplify interactive heavyweight computations. The segmentation process is seeded with an initial contour defined by the user in a 2D- projection (small blue curve in fig.1, rightmost vignette). The contour is then automatically adjusted to the region of interest, and vertically propagated and deformed upward and downward the slices. In our work, this part remains local, because its cost is low with respect to volume reconstruction; besides, remote segmentation falls into the RDRC scheme. In the facetization step, all surfaces joining two adjacent contours are generated as a set of triangles, which will be used for visualization and measurements. This part can be remotely computed. Interaction is at the beginning; also, the user can stop the reconstruction at any time.
One can object that LDRC is more of the cluster kind and do not really require a Grid, because data are not initially distributed, and the computation is a classical parallel one. While it is true that the hardware required is only a moderate size cluster, the key concept of Grids, service, virtual organizations and cross-domain access are present. The idea is to propose a high-end computing resource that can be accessed freely by any authorized organization, and which provides the appropriate QoS without the burden of maintaining the resource at the individual organization level.

2.3. Virtually overabundant resources

Fast computation requires parallel machines. The challenge comes from the following analysis. Today, an up-to-date dedicated cluster hardly provides acceptable response time. Better parallelization methods might help only to a limited extent: the algorithms considered are typically irregular, and the communication/computation ratio might be limiting very soon, considering the already fine grain of parallelism. With the evolution of the acquisition engines (multi-resolution scanners for instance), the increase in data size will effectively absorb the gains due to Moore's law. The obvious conclusion is that the grid computing resources must not be shared between users, and must be available on demand.
The first property is typical of high-end computing centres: a request is processed through a batch queue, but when granted, gets exclusive access to the allocated resources (at least processors, memory and the relevant segment of the fast network); the second property is typical of desktops, which are in general reserved for personal use. By pooling resources from multiple sites and applying operating system techniques like preemptive multitasking it should be feasible to create the illusion to users that there is a large computer dedicated to their use when in fact it is shared with multiple other users. This is of course the same concept as a shared departmental compute server, but generalized to the grid environment.

2.4. Scheduling issues

Scheduling is the general process of choosing where and when tasks are executed. Basically, the job defines its resource requirements [8], and the grid middleware matches them with its resources. Scheduling parallel tasks on a grid system naturally encompasses three levels.

- *Macro scheduling*. Each node (typically a cluster) of the grid features high-speed intra-node interconnections, and comparatively slow inter-node interconnection. Scheduling all the tasks of a parallel program on the same node exploits this locality. Scheduling tasks irrespective of locality exploits concurrency.
- *Queue scheduling*. When a job has been mapped, the current grid practice is to stage it in a queue until appropriate resources are made available. Defining what "appropriate" and "made available" mean depends on the overall grid policy, which is implemented through the policies of each node queuing system. Sophisticated

queue schedulers, like Maui or LSF, provide priority systems and availability definitions that can be finely configured using very many criteria.

- *Computer scheduling*. When the job is finally granted access to the computing resources, parallel scheduling deals with mapping and scheduling each task on processors; micro scheduling deals with time-sharing of hardware resources (CPU, memory, IO), and is usually delegated by the grid middleware to the usual OS time-sharing facilities.

To assess the costs and performance of some of these components, we have studied scenarios of progressive difficulty.

3. Parallel scheduling

We consider first the most simplistic scenario: the parallel machine is assumed to be fully available for exclusive use, without mapping and queuing costs. This scenario assesses the necessary condition for a compute-intensive algorithm to take advantage of grid facilities, that is to be amenable to parallelization, and highlights the parallel scheduling component. We have shown in a previous paper [9] that it is indeed the case for the PTM3D volume reconstruction algorithm (VRA in the following). This section briefly recalls the parallelization scheme, and provides extended performance results.

From the parallelization point of view, the VRA is master-slave category: it can be divided in a fairly large number of independent tasks, with a small input and a large output. While static (preplanned) parallel scheduling has been extensively studied (for a review of recent advances, see [11]), VRA can take little profit of this research, because nothing can be known in advance about the requirements of the computation. The number of tasks is unknown, because the user may stop the reconstruction at any time; the cost of each task cannot be accurately predicted from the input data.

To accommodate such an extremely dynamic profile, we use centralized self-scheduling. Self- scheduling requires two types of agents: one scheduling agent, and worker agents. The scheduling agent dispatches work to the workers following an on-demand scheme: when a worker finishes its work, it asks the scheduler for a new one (a work unit is one pair of adjacent slices). Besides performance, the scheduler/worker scheme naturally fits with interactivity. The flow of input data may be altered on the fly by some user interaction on the front-end, typically stopping or reshaping the segmentation.

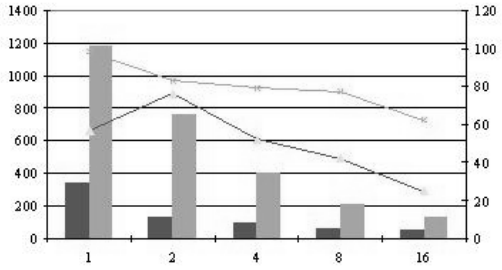

Figure3. Parallel scheduling

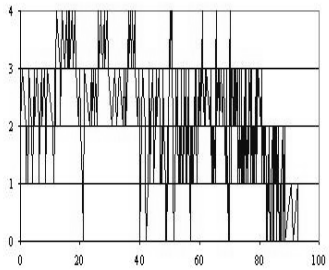

Figure4(a): Activity profile lungs 4 procs

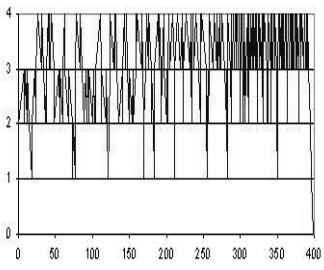

Figure4(a): Activity profile body 4 procs

Fig. 3 shows the performance of this scheme. The experimental setup is the LRI cluster (AMD 1800+, Linux 2.4.20, Ethernet 100 switch), the PTM3D front end on a desktop at LRI. The histogram records the execution time in seconds (left axis), and the curves the percentage of processor occupation averaged over all processors(right axis). The two examples are: e1 - two lungs (225 slices) and e2 - a body as in fig. 2, left size (156 slices). The body volume requires much more triangles than the lungs. The main result is that, for both cases, interactive time can be reached with moderate parallelism. A more detailed study shows that the speedup is linear up to 4 (resp 8) processors for e1 (resp e2), and significant with more processors for the heaviest computation (body).

As shown by average occupation, the scheduler is not able to keep all the processors busy. The detailed curves of fig. 4 showing the number of active processor along execution give a more precise view of this issue. This comes from three factors: network contention on the frontend-scheduler link; network contention on the cluster network; slackness between the fronted and the overall cluster when the segmentation does not generate slices fast enough. The superior performance of e2 comes from the better computation/input ratio, along with the shape of the occupancy curves, gives indication that the dominant factor is the last one.

4. Grid protocols

In this scenario, we want to assess the mere penalty of grid access. The grid middleware is Globus. Again, the performance criterium is the completion time (makespan) of a job, with exclusive access. The difference with the previous scenario is twofold: first, the latency of WAN access; second the penalty of grid protocols. Three implementations types have been considered.

- *Gliding* The scheduler agent and some worker agents are pre-launched on the remote resource; this scenario is identical to the the parallel scheduler one, except from the WAN latency. We borrow the gliding name from the Condor-Grid interface [10].
- *Regular Static* On activation of the PTM3D reconstruction tool by the user (clicking a button), a Globus job, including a scheduler and some workers, is submitted through a *globus_job_run* command. The scheduler and the workers communicate either through TCP channels or by the base Globus communication facilities (*globus_gram_myjob* functions). Static refers to the fact that the number of workers is part of the job specification and cannot be changed.
- *Regular Dynamic* The scheduler agent is kept local and launches one Globus job per slice, or group of slices.

A technically difficult problem in Globus 2 is the communication between the Globus resources and the outside world. Our current experiments use TCP; in a more realistic setting, firewalls at the two sites would probably block these communications. Substituting TCP channel with SSH tunnels will increases security; if the two sites are not willing to admit these, Globus provides many tools to communicate through files. Besides, Globus 3 offers extended tools.

The experiments were performed on the TAG Grid at ICPS, running Globus 2.4. The processors are AMD 2000+ and Pentium III 800MHz. The regular dynamic scheme is attractive, because it allows for adaptation to a dynamic behavior of the grid resources (see next section). However, the overhead of launching and terminating a job in the Globus framework is not negligible, in the order of 40s on a local grid. Thus, the regular dynamic scheme cannot be used at the application level. Experiments for the Regular Static scheme gave results very similar to the parallel scheduling scenario, which was expected: the Globus startup cost is paid only once.

5. Resource sharing

The scenarios considered so far assume the unrealistic hypothesis of exclusive access. In practice, an interactive job will have to share the grid resources with other processes. Interactive jobs are thus subject to regulation through the grid queuing policy like batch ones. For simplicity, processes will be coarsely categorized as either interactive or long; only the first category has real-time requirements. This section explores a quantitative assessment of the relationships between processing resources, interactive and non-interactive load. The overall management of queuing and micro-scheduling should ensure the following contracts.

- The real-time requirements of the interactive jobs are met.
- The delays incurred by the long jobs because of the interactive ones remain bounded
- Interactive jobs do not degrade resource utilization.
- Interactive jobs can be smoothly integrated with any existing queuing policies governing the long jobs.

The first two requirements target makespan, a user requirement. The third and fourth ones target sociologic acceptance. Grid resources (hardware, software, manpower) are not yet commodity resources as desktop PCs are, meaning that the funding organizations ask for some way of measuring the efficiency of their investment. The simplest and most frequent one (if not always meaningful) is resource utilization. Site managers design complex queuing policies in order to balance the requirements of different classes of customers amongst themselves and versus resource utilization. Because of the current structure of grid funding, such policies are often designed with long jobs in mind. Bootstrapping the interactive use of grids will not be possible if it requires a complete redesign of the sites policies, thus the fourth requirement.

The first question is the need for a *preemptive policy*. In a non-preemptive scheme, upon arrival in the queue all jobs, whether interactive or long, remains queued as long as the currently running job is not finished. Without preemption, but with higher priority for interactive jobs, queuing models give a quantitative evaluation of the average delay due to the job found in service upon arrival as $(\lambda_1 E(X_1^2) + \lambda_2 E(X_2^2))/2$, (Pollaczek-Kinchin mean value formula), where jobs from priority group p arrive in a Poisson stream with parameter λ_p and the service times for each group is the random variables X_p. The interactive jobs must thus be allowed to preempt resources from the long ones.

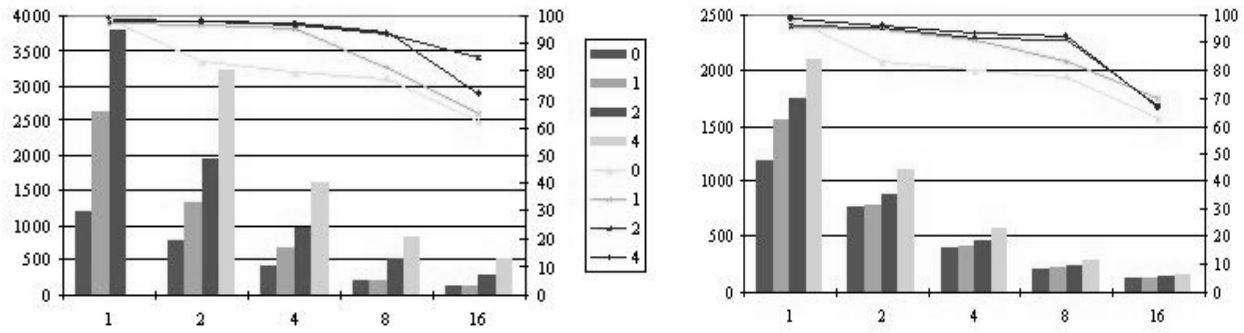

Figure 5: Impact of processor sharing

We first consider the scenario where only one interactive job contends with long jobs at a given time. This interactive job is given the resources necessary to complete within its time bounds, assuming that it runs concurrently with the long jobs prescribed by the long job policy at the same time. The grid scheduler defines what should be these resources from the parallelism requirement of the interactive job on one hand, and from its long job policy on the other hand. The adjustment function depends in a complex way on the algorithm and long job load, as exemplified in fig.6. The left graph displays the execution times of our examples against a background of $K = 1$, 2 or 4 running long jobs, all with equal processor priority (1 processor, 4 background jobs is not shown, being too high). It appears that one long job has minimal impact; four long jobs are acceptable only if at least 16 processors can be allocated. If there are not enough resources for the interactive job to meet its deadline with no special processor priority, the running long job should get less processor priority (be niced), as shown in fig. 6, right graph. More complex and adaptive scheduling algorithms can be studied, for instance increasing the resources allocated as the deadline approaches. While designing the adjustment function might require benchmarking to design, in a production context the imaging algorithm will be standardized, thus limiting the effort to the initial setup of the policy

Finally, it may happen that the interactive job does not meet its deadline, maybe because of an erroneous manipulation of the interactive user. We propose a sharing strategy inspired from the classical Foreground/Background scheduling algorithm. In this case, the job is degraded to the long job status, with any penalty that the local policy may want to inflict upon it.

This algorithm attains the objectives considered at the beginning of this section.

- If the interactive job has deterministic speedup, the grid scheduler is able to meet the deadline, if enough resource are available, or to decline the request if not.
- The delays incurred by the long jobs are bounded above by a constant deterministic factor c. This delay holds for each individual long job; besides, the average delay is bounded above by a factor equal to $c\lambda_1$.
- Interactive jobs do not degrade resource utilization. As the scheduling algorithm for long jobs remains unmodified, resource utilization can only be increased.
- The grid scheduling algorithm for long jobs does not need any modification. The only adverse case is the one where exclusive processor use by one long job is allowed. This kind of scheduling is not compatible with interactive jobs.

So far, we have considered systems where the interactive jobs compete with long jobs, which is likely to be the case in the first testbeds. However, if widely adopted, it might become more natural with respect to the funding structure to have grid nodes dedicated to interactive use. The previous scheduling algorithm provides a graceful transition from the "majority long" to the "all interactive scheme", allowing to leverage the initial experimental systems to exploitation ones without extensive redesign.

6. Conclusion and future work

Transparent access to computing power (including data access) is the founding grid metaphor. Seamless integration of the grid power to the everyday desktop use, when the desktop is insufficient, requires to revisit, not the grid concepts, but many trends in grid software. Departing from the computing centre model means to put emphasis not on work throughput, but on response time at all levels of the grid protocols, while retaining the key advances of Grid technology allowing to build virtual organizations: security, unique login, cross-domain access etc. This paper shows that a combination of application-level schedulers and a moderately intrusive queuing policy can provide true (guaranteed) QoS for an interactive application with fine-grain parallelism and normal exploitation of the leftover resources. Future work will consider adaptive resource allocation algorithms, and the specification of a grid-enabled dynamic scheduler as an independent tool .

Acknowledgments

We thank Stéphane Génaud and the TAG team, who have given us access to the Globus infrastructure at ICPS, and greatly helped us in setting up the experiments.

Bibliography

[1] A. Osorio, P-J. Valette, A. Mihalcea, J.Atif and X. Ripoche. A new PC based software to perform semi-automatic hepatic segmentation using CT and MR images. *InfoRAD 2002, RSNA'02*

[2] R. Stefanescu, X. Pennec and N. Ayache. Grid Enabled Non-Rigid Registration with a Dense Transformation and A Priori Information. In *Proc. of MICCAI'03*, LNCS, November 2003. Springer Verlag.

[3] V. Breton, R. Medina, J. Montagnat. DataGrid, Prototype of a BioMedical Grid. Methods MIMST, 42(2), pp 143-148, 2003

[4] S. Santini S, A. Gupta A. The role of Internet Imaging in the Biomedical Informatics Research Network. In *Procs of SPIE*, Vol. 5018 . 2003

[5] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, and B. Tierney. File and Object Replication in Data Grids. *Journal of Cluster Computing*, 5(3), pp 305-314, 2002.

[6] J. S. Plank, S. Atchley, Y. Ding and M. Beck. Algorithms for High Performance, Wide-Area Distributed File Downloads. *Parallel Processing Letters*, 13(2), pp. 207-224, 2003

M. Beck, T. Moore and J. S. Plank. An End to End approach to Globally Scalable Programmable Networking. FDNA'03 at ACM SIGCOMM 2003
[7] IMAGE`03: Images, Medical Analysis and Grid Environments. To appear in UK e-Science Technical Report Series. http://www.nesc.ac.uk/technical_papers/
[8] R. Raman, M. Livny, and M. Solomon. Resource Management through Multilateral Matchmaking. *Procs 9th IEEE Symp. on High Performance Distributed Computing (HPDC9)*, pp 290-291. 2000.
[9] C. Germain, A. Osorio and R. Texier. A case study in medical imaging and the Grid. 1st Health Grid Conference. pp 110-118. january 2003. EC-IST.
[10] D. Thain et al. Pipeline and Batch Sharing in Grid Workloads. In *Procs 12th IEEE Symposium on High Performance Distributed Computing*. 2003.
[11] O. Beaumont, A. Legrand and Y. Robert. Optimal algorithms for scheduling divisible workloads on heterogeneous systems. Procs 12th IEEE Heterogeneous Computing Workshop. 2003